

每周工作汇报

姓名	侯宇轩	日期	2019.6.26-2019.6.30
----	-----	----	---------------------

1. 本周工作

0. 时间线（标黄的为现在正在处理，标绿为已完成）

第一阶段：测试数据+CPU

1.1 准备一个浮点数测试数据。（目前小鼠数据是灰度值数据，可以直接将 90M 测试集转为浮点数先用）（发现可能不需要浮点数，直接使用整数（灰度值）的数据转为 YUV 格式）

1.2. 将其在 CPU 上转换为 YUV 4: 2: 0 格式。（目前的方法：直接修改扩展名）
转换时通过补齐片数到 3 的倍数修正了 YUV420 格式片数不是整数片的问题

1.3. 将得到的 YUV 格式数据在 CPU 上转换为 H.264 格式。（使用他人的代码，已找到）

1.4 将得到的 H.264 格式解码为 YUV 格式。（使用他人的代码，已找到）
decode 时出现的丢帧情况已解决

1.5 将 YUV 格式转回为浮点数测试数据。（目前的方法：直接修改扩展名）

1.6 进行比较：使用压缩解压后数据/未压缩数据造成的渲染质量的不同。

其中，1.1-1.5 需要约一周时间（6.14-6.21），1.6 需要 2 天时间（6.22,6.24）

第二阶段：测试数据+GPU

2.1 寻找 1.4 步骤的代替，设法在 GPU 上将 H.264 格式转回为 YUV 格式(Decoder)。

目前，根据任老师推荐的 Nvidia Decoder Codec SDK, 细分任务如下：

2.1.1 安装该 SDK, 运行其例子，了解其工作流程

2.1.2 编写调用 Decoder 的代码

2.1.3 使用小的真数据进行测试 Decoder（若不行，可以再换假数据测试）

2.1.4 使用更大的数据测试 Decoder，最大：4096*4096*4096

2.2 进行同 1.6 的测试。

2.3 将浮点数->YUV->H.264->YUV->浮点数的流程组成完整的代码。

2.4 寻找 1.3 步骤的代替，设法在 GPU 上将 YUV 格式转为 H.264 格式(Encoder)。

其中，2.1 需要两周时间，(6.24-7.8)

2.2、2.3 需要一周时间（7.9-7.15）。

2.4 需要一周时间。（7.16-7.22）

第三阶段：大数据+GPU

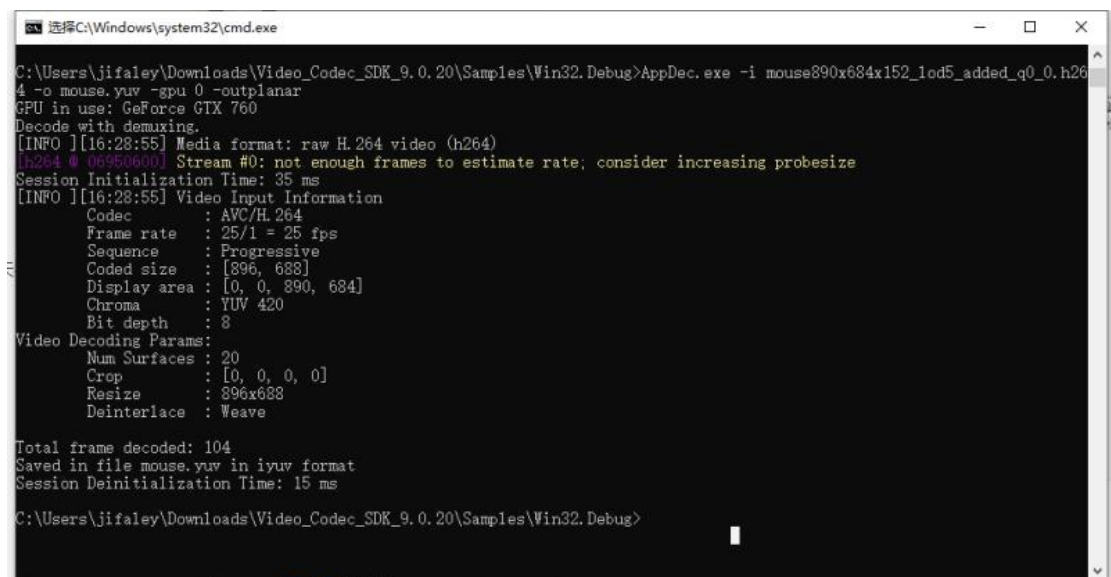
3.1 设计如何分块处理大数据.....

2. 本周工作：

完成了 2.3.1-2.3.3， 安装了 Nvidia Codec SDK，编写了调用 Decoder 的代码，使用小的真数据测试了 Decoder。

将之前使用 CPU Encoder 编码的 H264 文件，调用 GPU Decode:

（数据：下采样 5 次的真数据，大小 890*684*152，padding 成为 890*684*146，原始数据大小 90M）



```
C:\Users\jifaley\Downloads\Video_Codec_SDK_9.0.20\Samples\Win32.Debug>AppDec.exe -i mouse890x684x152_1cd5_added_q0_0.h264 -o mouse.yuv -gpu 0 -outplanar
GPU in use: GeForce GTX 760
Decode with demuxing.
[INFO ][16:28:55] Media format: raw H.264 video (h264)
[h264 @ 06950600] Stream #0: not enough frames to estimate rate; consider increasing probesize
Session Initialization Time: 35 ms
[INFO ][16:28:55] Video Input Information
    Codec       : AVC/H.264
    Frame rate  : 25/1 = 25 fps
    Sequence   : Progressive
    Coded size  : [896, 688]
    Display area : [0, 0, 890, 684]
    Chroma     : YUV 420
    Bit depth  : 8
Video Decoding Params:
    Num Surfaces : 20
    Crop         : [0, 0, 0, 0]
    Resize       : 896x688
    Deinterlace  : Weave
Total frame decoded: 104
Saved in file mouse.yuv in iyu format
Session Deinitialization Time: 15 ms
C:\Users\jifaley\Downloads\Video_Codec_SDK_9.0.20\Samples\Win32.Debug>
```

对量化参数 $q=0$ （最高质量 Encoding）的 Decode 结果如下：

mouse890x684x152_lod5_added_q0_0.h264	2019/6/21 16:45	H264 文件	14,829 KB
mouse890x684x152_lod5_added_q0_0_decoded	2019/6/21 16:48	RAW 文件	92,741 KB
mouse890x684x152_lod5_added_q0_0_decoded_gpu	2019/6/29 16:20	RAW 文件	92,741 KB

第一个是用于解码的 H264 文件（大小 14M），第二个是 CPU 解码结果，第三个是 GPU 解码结果。首先我们可以看到两种方式解码后文件大小完全相同。

在 Ray-caster 中观察其区别：



左图：CPU 解码结果。右图：GPU 解码结果。

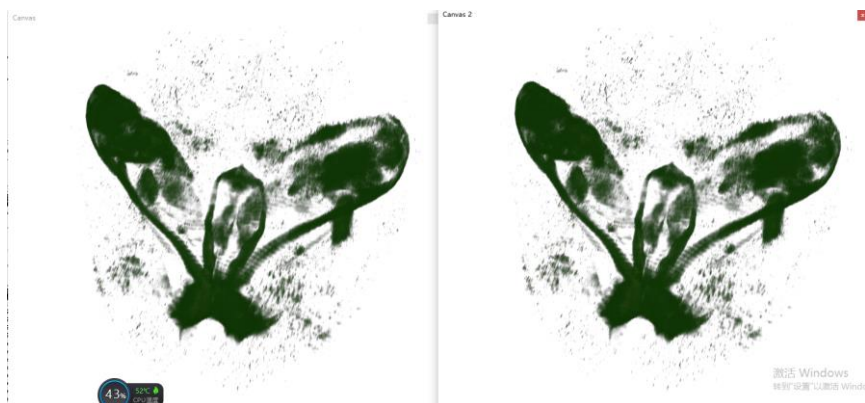
可以看到，基本没有区别。

之后，对之前选定的质量-大小折中量化参数 $q=30$ 下 编码的数据进行解码：

mouse890x684x152_lod5_added_q0_30.h264	2019/6/21 16:46	H264 文件	860 KB
mouse890x684x152_lod5_added_q0_30_decoded	2019/6/21 16:49	RAW 文件	92,741 KB
mouse890x684x152_lod5_added_q0_30_decoded_gpu	2019/6/29 16:58	RAW 文件	92,741 KB

此时 H264 文件大小为 0.86M，比原来的 90M 压缩近 100 倍。

质量比较如下：



左图：CPU 解码结果 右图：GPU 解码结果

可以看到，还是基本没有区别。那么可以说明 GPU 解码也是正确而可行的。

2. 明日计划

继续 2.1.4 与 2.2，在更大的数据上测试。